УДК 519.8

І.В. СЕРГІЄНКО
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *Sergiyenko@nas.gov.ua.*

В.П. ШИЛО
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *v.shylo@gmail.com*.

В.О. РОЩИН
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *dopt135@gmail.com*.

ОБ'ЄДНАННЯ АЛГОРИТМІВ ДЛЯ РОЗВ'ЯЗАННЯ ДИСКРЕТНИХ ОПТИМІЗАЦІЙНИХ ЗАДАЧ[1]

Анотація. Розглянуто об'єднання (портфелі і команди) оптимізаційних алгоритмів, їхні характеристики та вплив на прискорення процесу оптимізації. Досліджено застосування об'єднань алгоритмів глобального рівноважного пошуку до окремих задач дискретної оптимізації, різні схеми обміну інформацією в командах алгоритмів. Значну увагу приділено експериментальному дослідженню розроблених портфелів і команд алгоритмів, проведеному як у режимі реального часу, так і з використанням багатопроцесорного обчислювального комплексу СКІТ-4 Інституту кібернетики ім. В.М. Глушкова НАН України, та їхньому порівняльному аналізу. Отримано оцінки ефективності об'єднань алгоритмів, згідно з якими командний підхід має значні переваги над портфелями алгоритмів.

Ключові слова: об'єднання (портфелі і команди) алгоритмів, дискретна оптимізація, обмін інформацією, експериментальні дослідження, ефективність об'єднань алгоритмів.

ВСТУП

Побудова сучасного інформаційного суспільства потребує використання нових комп'ютерних технологій [1] для розв'язання численних задач аналізу та оптимізації процесів прийняття рішень у різноманітних галузях науки та практики. Ці задачі виникають під час бюджетного і макроекономічного планування та прогнозування, прийняття управлінських рішень, проєктування складних технічних систем і мереж тощо. Важливим напрямом досліджень, пов'язаним зі створенням наукового фундаменту в галузі побудови сучасних комп'ютерних технологій, є методи оптимізації, зокрема дискретної оптимізації, наприклад [2–12].

Клас задач дискретної оптимізації дуже різноманітний. Ця сфера досліджень цікава з математичного погляду, оскільки пов'язана з проблемами комбінаторної та дискретної математики. Багато питань у них пов'язано з перебором, переліченням (повним або частковим) варіантів — допустимих розв'язків задачі. Збільшення кількості прикладних проблем, що описуються математичними моделями дискретної оптимізації, і бурхливий розвиток обчислювальної техніки, який розширює можливості перебору варіантів, викликали значний інтерес до таких задач. Більшість задач дискретної оптимізації є універсальними (NP-важкими) задачами. За постійно зростаючої складності об'єктів, що оптимізуються, природним є й ускладнення їхніх математичних моделей. Це значно утруднює пошук розв'язків таких задач.

Під час розв'язання дискретних оптимізаційних задач великої розмірності виникає потреба в обробленні великих обсягів інформації за прийнятний час. Для ефективного та часто застосовуваного способу розв'язання цих задач

[1]Роботу виконано за часткової підтримки гранту 2021.01/0136 Національного фонду досліджень України.

доцільно застосовувати багатопроцесорні обчислювальні комплекси. Паралельні обчислювальні системи надають фахівцям інструменти для розв'язання оптимізаційних задач безпрецедентного масштабу. Проте без ефективних і масштабованих паралельних методів не можна використовувати ці обчислювальні ресурси. У зв'язку з цим важливу роль відіграють розроблені авторами об'єднання (портфелі і команди) алгоритмів для розпаралелювання процесу розв'язання задач дискретної оптимізації. Крім того, можливість швидкого оброблення великих масивів даних дає змогу розв'язувати реальні задачі в економіці, космічній галузі, медицині, біології [13] тощо.

У цій роботі представлено деякі підсумкові результати проведених досліджень перспективного наукового напряму, пов'язаного з розпаралелюванням обчислень. Розвиток паралельних алгоритмів розв'язання складних дискретних оптимізаційних задач зумовлений як нагальними потребами економічного і технічного характеру, так і необхідністю подальшого розвитку теорії оптимальних рішень.

РОЗПАРАЛЕЛЮВАННЯ ПРОЦЕСУ ОПТИМІЗАЦІЇ ЗА ДОПОМОГОЮ ОБ'ЄДНАНЬ АЛГОРИТМІВ

Під час розроблення алгоритмів з паралельною організацією обчислень слід ураховувати специфіку розпаралелювання і не дотримуватися стандартів послідовних алгоритмів. Послідовний і паралельний варіанти — це різні алгоритми, що мають різну мету і тому немає сенсу порівнювати їхню роботу у послідовному режимі.

Припустимо, що $A = \{A_1, \dots, A_m\}$ — множина алгоритмів і доступні p логічних процесорів $\{C_1, \dots, C_p\}$. Під логічним процесором будемо розуміти або один фізичний процесор, або одне ядро фізичного процесора, або один фізичний потік ядра процесора. Кожен логічний процесор повинен використовуватися одним алгоритмом множини A. Крім того, один і той самий алгоритм можна застосовувати на різних логічних процесорах. Останній випадок має сенс, коли мова йде про рандомізований алгоритм розв'язання дискретної оптимізаційної задачі, випадкову поведінку якого визначає датчик псевдовипадкових чисел. Копією такого алгоритму будемо називати його варіант, одержаний для одного початкового значення цього датчика. Очевидно, що за різних початкових значень датчика створюються різні копії вихідного алгоритму, які дають змогу здійснювати пошук відмінних між собою розв'язків.

Об'єднанням алгоритмів назвемо деяку підмножину множини A алгоритмів, які працюють паралельно над розв'язанням однієї задачі. Цю підмножину задають списком union list $\{n_1 A_1, \dots, n_m A_m\}$ алгоритмів A_i із зазначенням кількості n_i, $i = 1, \dots, m$, використовуваних ними логічних процесорів, причому $\sum_{i=1}^{m} n_i = p$. Цей список може бути або статичним (не змінюватися під час розв'язання задачі), або динамічним (змінюватися у певні з початку розв'язання моменти часу), або адаптивним (зміни у списку будуть залежати від ходу розв'язання оптимізаційної задачі).

Об'єднання алгоритмів, які не обмінюються інформацією і працюють незалежно один від одного, називають портфелем алгоритмів. Хоча алгоритми портфеля працюють незалежно один від одного, для зручного оброблення отримані ними результати варто зберігати в одному загальнодоступному місці. Після завершення роботи всіх алгоритмів $A_1, \dots, A_m$ портфеля вибирають найкращий з отриманих розв'язків задачі. Поняття портфеля алгоритмів запозичене з галузі фінансів, де інвестори об'єднують різні активи для зниження ризику. Згідно

з цією аналогією назва «портфель алгоритмів» передбачає, що немає зв'язку або співробітництва між учасниками портфеля, оскільки немає чіткого співробітництва між активами у фінансових портфелях.

Час розв'язання задачі за допомогою портфеля чи команди алгоритмів, які позначимо як *unit*, визначається часом найшвидшого її розв'язання на одному із процесорів, тобто $t_{unit} = \min\limits_{i=1,\dots,p} t_i$, де t_i — час розв'язання задачі на i-му процесорі.

Ранні теоретичні дослідження паралельних портфелів алгоритмів пов'язані з незалежними випадковими алгоритмами оптимізації. Наявність випадкових кроків є невід'ємною частиною їхньої логіки. Згідно з [5, 14, 15] паралельне виконання незалежних копій випадкового алгоритму часто забезпечує ефективні стратегії прискорення.

Під час розроблення портфеля алгоритмів можна паралельно запускати декілька копій одного й того самого алгоритму (однорідний портфель алгоритмів) або комбінувати об'єднання алгоритмів (неоднорідний портфель алгоритмів). У загальному випадку неоднорідний портфель кращий за однорідний. Дослідженню портфелів алгоритмів присвячено низку робіт, наприклад [14–21].

Хоча в літературі зазначено, що однорідні портфелі не є оптимальними, ці портфелі набагато легше налаштовувати, ніж неоднорідні. Якщо розрив між ефективністю неоднорідного й однорідного портфелів не є істотним, простота налаштування є, як правило, більш кращим варіантом для вибору типу портфеля.

Перспективним є підхід до розпаралелювання копій імовірнісного алгоритму (однорідний портфель) з рестарт-розподілами [5]. Рестарт-розподіл — це розподіл часу розв'язання задачі, за якого можна покращити якість її розв'язання шляхом використання деякої процедури перезапуску алгоритму. Важливо, що в багатьох випадках у разі застосування цього підходу задачу розв'язують набагато швидше, ніж із використанням інших видів розпаралелювання. Цього можна досягти, наприклад, у випадку, коли розподіл часу розв'язання задачі алгоритмом є рестарт-розподілом. Властивості розподілу часу розв'язання залежать як від розв'язуваної задачі, так і від застосовуваного алгоритму. Якщо для оптимізаційного алгоритму використовують, наприклад, оптимальне значення рестарт-критерію, то час розв'язання задачі цим алгоритмом вже не буде рестарт-розподілом.

У разі розпаралелювання копій алгоритму є два можливих сценарії — використовувати рестарт для всіх процесорів чи ні. У загальному випадку вигідніше застосовувати різні рестарт-критерії зупинки алгоритму для різних процесорів. Однак, через складність визначення цих величин для великої кількості процесорів зазвичай використовують одне значення рестарт-критерію для всіх процесорів.

Припустимо, що є набір доступних алгоритмів $A = \{A_1, \dots, A_m\}$ і набір доступних логічних процесорів $C = \{C_1, \dots, C_p\}$. Нехай ξ_i — час виконання алгоритму A_i. Задачу вибору неоднорідного портфеля алгоритмів для мінімізації загального середнього часу розв'язання можна записати так: знайти

$$\min \{f^{het} = E\,[\min(\xi_{i_1}, \dots, \xi_{i_p})]\} \tag{1}$$

за умови

$$1 \le i_j \le m,\ \ j = 1, \dots, p, \tag{2}$$

де i_j — номер алгоритму, що виконується на логічному процесорі j, E — математичне сподівання.

Як варіант, можна сформувати портфель на основі декількох копій одного й того самого алгоритму. Запишемо відповідну задачу вибору однорідного портфеля алгоритмів у такому вигляді: знайти

$$\min \{f^{hot} = E[\min (\xi_{i_1}, \ldots, \xi_{i_p})]\} \tag{3}$$

за умови

$$i_1 = \ldots = i_p. \tag{4}$$

Зрозуміло, що задача (1), (2) набагато складніша за задачу (3), (4). Кількість можливих розв'язків задачі (1), (2) дорівнює m^p, а кількість можливих розв'язків задачі (3), (4) — m. Очевидно, що оптимальний розв'язок задачі (1), (2) кращий ніж оптимальний розв'язок задачі (3), (4), оскільки допустима область першої включає допустиму область останньої, тоді як обидві задачі мають однакову цільову функцію.

Досліджено [14, 21], що кращий неоднорідний портфель алгоритмів у кращому випадку працює в 1.58 рази швидше, ніж кращий однорідний портфель алгоритмів за умови, що окремі алгоритми є рестарт-алгоритмами з оптимальним часом рестарту.

Підмножину множини A алгоритмів об'єднання, які обмінюються інформацією, називають командою алгоритмів.

Одним із можливих шляхів прискорення процесу розв'язання задач дискретної оптимізації у разі розпаралелювання алгоритмів команд є їхня взаємодія, обмін отриманою інформацією між ними. Під час обміну інформацією можна посилити кращі якості алгоритмів. Це зумовить зменшення часу їхньої роботи і покращення якості розв'язків.

Алгоритми команди не тільки обмінюються інформацією один з одним, але можуть включати алгоритм для реструктуризації команди, модифікації обміну інформацією, ґрунтуючись на ході процесу оптимізації. Для підвищення ефективності команда алгоритмів повинна містити процедури, що можуть використовувати зовнішню інформацію, яка передається всередину команди. Наприклад, якщо задача розв'язується за допомогою команди стандартних алгоритмів локального пошуку, які ініціюються незалежно згенерованими початковими розв'язками, то обмін інформацією не має сенсу, оскільки інформація, яка передається, не може регулювати кроки локального пошуку.

Питання про інформацію є ключовим. Якою інформацією достатньо обмінюватися алгоритмам команди, щоб досягти найкращого результату, і як часто її потрібно передавати? Взагалі кажучи, питання про інформацію досить нетривіальне. Неправильно думати, що обмін будь-якою інформацією корисний. Для певних алгоритмів необхідно забувати, табувати отриману інформацію. Надлишок інформації, що передається, може бути так само шкідливим, як і відсутність обміну інформацією.

Алгоритм повинен насамперед «уміти» сприймати інформацію, використовувати її. Наприклад, під час розв'язання задач за допомогою алгоритмів табу будь-який обмін інформацією між ними не має сенсу, оскільки ці алгоритми не дуже пристосовані до використання зовнішньої інформації. У цьому випадку треба додатково використовувати або якийсь координувальний алгоритм, який буде накопичувати інформацію, одержувану цими алгоритмами, і «керувати» їхнім пошуком або поєднувати ці алгоритми зі схемами випадкового збурення (ітерованості).

Алгоритм глобального рівноважного пошуку (ГРП) [5, 6] і генетичний алгоритм здатні ефективно обмінюватися інформацією. Генетичні алгоритми можуть обмінюватися, зокрема, представниками своїх популяцій. Це може покра-

щити якість знайдених розв'язків і швидкість їхнього одержання. В алгоритмі глобального рівноважного пошуку імовірності випадкового пошуку адаптивно налаштовуються, використовуючи отримані розв'язки. Тому обмін такими розв'язками між копіями цього алгоритму дасть змогу прискорити процес одержання якісних розв'язків.

Метод ГРП має високу обчислювальну ефективність у багатьох застосуваннях, проте однією з найважливіших його властивостей є здатність бути ефективним командним алгоритмом. Це досягається за допомогою вбудованих структур пам'яті, які можуть обробляти розв'язки, отримані будь-яким іншим алгоритмом. Визначення хорошого протоколу обміну інформацією не є тривіальним. У деяких випадках обмін інформацією може перешкоджати роботі команди і його варто уникати.

Найкращий з усіх допустимих розв'язків задачі, розглянутих до цього кроку обчислювального процесу, будемо називати рекордним, а відповідне йому значення цільової функції — рекордом.

Далі під час дослідження команд алгоритмів будемо розглядати просту стратегію обміну інформацією, де тільки рекордний розв'язок і відповідний йому рекорд є предметами обміну.

ЗАСТОСУВАННЯ ОБ'ЄДНАНЬ АЛГОРИТМІВ ГЛОБАЛЬНОГО РІВНОВАЖНОГО ПОШУКУ

Ймовірнісний метод глобального рівноважного пошуку добре зарекомендував себе під час розв'язання на ЕОМ багатьох класів дискретних оптимізаційних задач у послідовному та паралельному режимах (див., наприклад, [6, 20, 22, 23]). Його вважають одним із кращих сучасних наближених методів булевого програмування. Зокрема, розроблено і досліджено послідовні і паралельні алгоритми ГРП для задачі булевого квадратичного програмування без обмежень (UBQP) вигляду [6]: знайти

$$\max\left\{ f(x)=\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j \,|\, x\in B^n \right\}, \tag{5}$$

де q_{ij} — елементи симетричної дійсної матриці Q порядку n, B^n — множина n-вимірних векторів з компонентами 0 або 1. Задача (5), крім важливих практичних застосувань у фізиці, хімії, інженерії, медицині тощо, також є хорошим «полігоном» для перевірки різних оптимізаційних підходів. Для різних наборів даних її цільова функція породжує різноманітні оптимізаційні «ландшафти».

Можна виділити два основні типи структур локальних оптимумів задачі (5). Перший тип стосується множини локальних оптимумів, зосереджених у досить компактній, окремій області. Якщо алгоритм ГРП знаходить розв'язки в цій області, то він досить швидко знаходить у ній і локальний оптимум з найбільшим значенням цільової функції. Другий тип можна розглядати як об'єднання структур локальних оптимумів першого типу, причому їхні області примикають одна до одної, створюючи деяку протяжну область.

У термінах булевого квадратичного програмування можна представити деякі задачі теорії графів, зокрема, задачу про максимальний зважений розріз графу (WMAXCUT). Її математична модель є такою: знайти

$$\max\left\{ f(x)=\sum_{(i,j)\in E} w_{ij}(x_i - x_j)^2 \,|\, x\in B^n \right\},$$

де w_{ij} — вага, яка відповідає кожному ребру (i, j) графу G множини E його ребер. Задача WMAXCUT — це класична проблема дискретної оптимізації з

численними практичними застосуваннями [24–26]. Серед них слід назвати проєктування мереж зв'язку, надвеликих інтегральних схем, фазованих антенних решіток, моделювання нейронних мереж, розпізнавання образів, аналіз великих масивів даних, задачі фізики твердого тіла тощо. Для цієї задачі на основі методу ГРП та його модифікацій розроблено і досліджено конкурентоздатні послідовні і паралельні алгоритми [6, 20, 22, 23 та ін.].

Локальні оптимуми першого типу часто зустрічаються в задачах UBQP. Так, усі тестові приклади із роботи [27] для задач UBQP мають цю структуру локальних оптимумів. Крім того, тестові задачі MAXCUT G1–G54 [28] також мають структуру локальних оптимумів першого типу, а тестові задачі MAXCUT G55–G81 [28] мають уже структуру локальних оптимумів другого типу.

Очевидно, що структуру оптимумів досліджуваної задачі слід враховувати під час побудови команд алгоритмів для її розв'язання. Якщо задача має структуру локальних оптимумів першого типу, то головна проблема полягає у правильній диверсифікації пошуку, використанні інформації, отриманої під час розв'язання задачі. Деякому алгоритму команди не потрібно досліджувати структури локальних оптимумів першого типу, які уже, можливо, розглянуті цим або іншими алгоритмами команди. З іншого боку, якщо задача має структуру локальних оптимумів другого типу, то головною проблемою є правильна інтенсифікація пошуку, використання інформації, отриманої під час розв'язання задачі. Дослідження всіма алгоритмами команди певної структури локальних оптимумів другого типу суттєво зменшує обчислювальний час, потрібний для знаходження кращого локального оптимуму всієї структури другого типу.

Розглянемо схеми обміну інформацією в командах алгоритмів для дослідження структур локальних оптимумів першого та другого типів. Алгоритми команди team1 для структур першого типу обмінюються інформацією за такою схемою. Нехай $x^i(t)$ — рекордний розв'язок, знайдений алгоритмом A_i, $i = 1, \ldots, m$, команди в момент часу t, а $x^*(t)$ — рекордний загальний розв'язок, який зберігається у спільній пам'яті і доступний всім алгоритмам команди. Тоді $f(x^*(t)) = \max\limits_{i \in A} \{f(x^i(t))\}$, де A — множина алгоритмів команди. Кожного разу, коли алгоритм A_i команди знаходить рекордний розв'язок $x^i(t)$, для якого $f(x^i(t)) > f(x^*(t))$, покладаємо $x^*(t) = x^i(t)$. Якщо $f(x^i(t)) < f(x^*(t))$ для всіх алгоритмів A_i, то покладаємо $x^i(t) = x^*(t)$. Ці два правила визначають протокол обміну, який використовується командою team1.

Алгоритми команди team2 для структур локальних оптимумів другого типу застосовують схему обміну, подібну до team1, з однією відмінністю. Із результатів розв'язання задач за допомогою портфеля алгоритмів під час кожної спроби було відомо, який алгоритм знайшов найкращий розв'язок (назвемо його лідером). Команди алгоритмів team2 використовували таку схему в кожній спробі. У файл спільної пам'яті вносилися дані всіх алгоритмів команди, а інформація зчитувалася всіма алгоритмами, крім лідера. Очевидно, що результати роботи команди team2 принаймні не гірші, ніж отримані за допомогою портфеля алгоритмів.

Для обчислювальних експериментів використано недавно розроблені алгоритми ГРП (GES) нового покоління, названі GESPR, і відповідне програмне забезпечення. З їхньою допомогою розв'язано задачі про максимальний зважений розріз графу дуже великої розмірності. Характерною рисою алгоритмів GESPR є проведення осциляції навколо найкращого знайденого розв'язку на основі методології Path Relinking [29]. Як і в [30], експерименти проведено

з 71 тестовою задачею, що широко використовуються для перевірки алгоритмів розв'язання задач WMAXCUT. Ці задачі можна завантажити за посиланням http://www.stanford.edu/~yyye/yyye/Gset/. Вони включають тороїдальні, планарні та випадкові графи з кількістю вершин $|V|$ від 800 до 20 000 і значення ваги ребер 1, 0 або –1.

Експериментальні розрахунки з розпаралелювання процесу розв'язання задачі WMAXCUT проведено за допомогою портфеля і трьох команд копій чотирьох алгоритмів GESPR з використанням PC Intel®Core™ i7-3770 CPU @ 3.40 Ghz і 8.0 GB оперативної пам'яті в режимі реального часу. Інакше кажучи, всі алгоритми стартували одночасно.

Аналіз результатів проведених експериментів дає змогу дійти висновку, що процес оптимізації умовно можна розділити на три етапи: пошук розв'язків поганої, середньої і хорошої якості. На першому етапі, коли ще немає небезпеки потрапити в западню низькоякісних розв'язків, варто проводити активний обмін інформацією між алгоритмами команди. Цей обмін допоможе швидко перейти до другого етапу, де є більша ймовірність потрапити в западню. Щоб уникнути цього і забезпечити кращу диверсифікацію, на цьому етапі варто скоротити обмін інформацією. Нарешті, на третьому етапі треба знову підвищити рівень обміну інформацією, щоб активізувати пошук високоякісних розв'язків. Спробу здійснити ці ідеї реалізовано в команді алгоритмів team3.

Отримані результати експериментальних досліджень свідчать про високу обчислювальну ефективність алгоритму GESPR: портфель і команди алгоритмів GESPR знайшли кращі рекорди, ніж раніше відомі [30]. Важливо, що всі три команди перевершили портфель алгоритмів за якістю розв'язків. У табл. 1 наведено деякі результати порівняльного дослідження рекордів, знайдених командою алгоритмів GESPR і алгоритмом BLS [30]. Кожен алгоритм розв'язував задачу із роботи [28] 20 разів. У першому стовпчику табл. 1 вказано номер задачі, у другому — потужність $|V|$ множини V (кількість вершин графу G). Для всіх задач, наведених у табл. 1, алгоритми GESPR з обміном інформацією знайшли нові рекорди.

Експериментальні дослідження з розпаралелювання процесу розв'язання задач WMAXCUT проведено також з використанням багатопроцесорного обчислювального комплексу СКІТ-4 Інституту кібернетики ім. В.М. Глушкова НАН України. Розв'язано задачу G81 про максимальний зважений розріз графу з кількістю вершин 20000.

На рис. 1 показано прискорення процесу розв'язання задачі G81 за допомогою портфелів portfolio з 8, 16, 32, 128 алгоритмів і команд team з 8, 16, 32 алгоритмів GES порівняно з одним алгоритмом.

Таблиця 1. Результати порівняльного дослідження рекордів

Задача	$\|V\|$	BLS	GESPR	Задача	$\|V\|$	BLS	GESPR
G55	5000	10294	**10299**	G64	7000	8735	**8751**
G56	5000	4012	**4017**	G65	8000	5558	**5562**
G57	5000	3492	**3494**	G66	9000	6360	**6364**
G58	5000	19263	**19293**	G67	10000	6940	**6950**
G59	5000	6078	**6086**	G70	10000	9541	**9591**
G60	7000	14176	**14188**	G72	10000	6998	**7006**
G61	7000	5789	**5796**	G77	14000	9926	**9938**
G62	7000	4868	**4870**	G81	20000	14030	**14054**
G63	7000	26997	**27045**	—	—	—	—

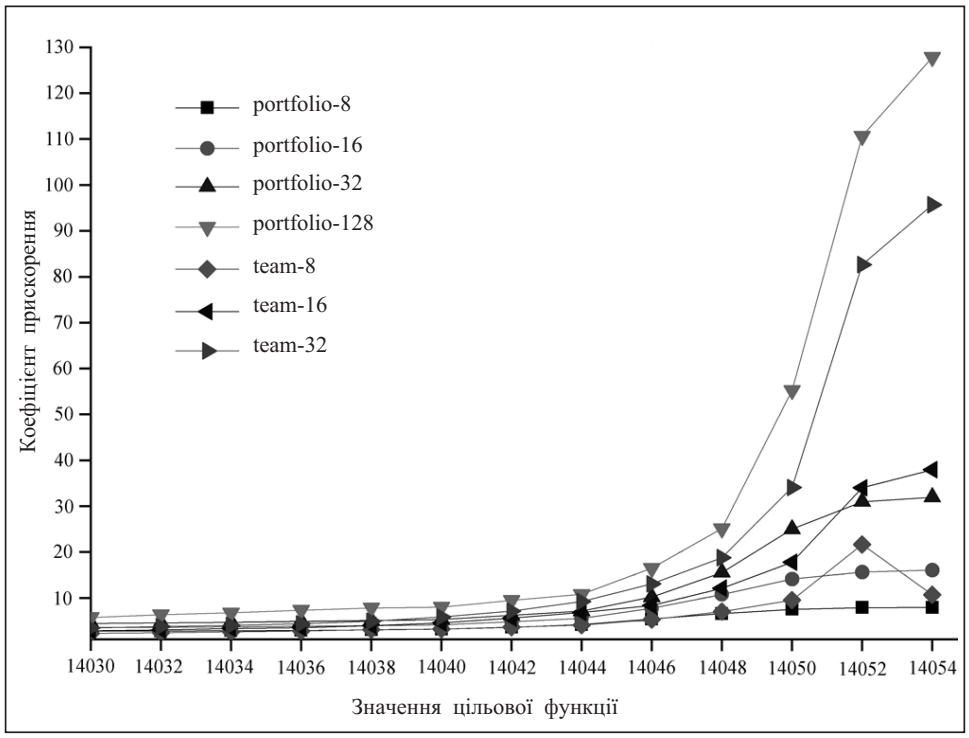

Рис. 1. Графіки прискорення процесу розв'язання задачі G81

Аналіз отриманих результатів експериментальних розрахунків свідчить про те, що портфель із 128 алгоритмів GES знаходить розв'язок задачі зі значенням цільової функції 14054 у 127 разів швидше, ніж один алгоритм. Команда з 32 алгоритмів знаходить розв'язок з таким самим значенням цільової функції у 95 разів швидше, ніж один алгоритм. Як бачимо, під час використання команд алгоритмів досягнуто надлінійне прискорення. Такі самі результати спостерігаються для команд, які складаються з восьми та шістнадцяти алгоритмів. Очевидно, що для портфеля з чотирьох команд алгоритмів, які містять по 32 алгоритми, можна одержати прискорення відносно одного алгоритму більше, ніж у 289 разів.

ОЦІНКИ ЕФЕКТИВНОСТІ КОНФІГУРАЦІЙ КОМАНД ТА ПОРТФЕЛІВ АЛГОРИТМІВ

Розглянуто три різні конфігурації однорідних портфелів portfolio-8, portfolio-16 і portfolio-32 з кількістю $N = 8, 16$ і 32 незалежних алгоритмів відповідно. Також досліджено три конфігурації команд team-8, team-16 і team-32 з кількістю алгоритмів 8, 16 і 32 відповідно. Конфігурації команди відповідали певному протоколу зв'язку. Крім того, оцінено ефективність 16 незалежних копій алгоритмів команди team-8, восьми незалежних копій алгоритмів team-16 і чотирьох незалежних копій алгоритмів team-32. Ці конфігурації позначено team-8x16, team-16x8 і team-32x4 відповідно.

Для оцінки ефективності конфігурацій портфеля виконано 2560 запусків незалежних алгоритмів тривалістю виконання 20 годин кожен. Для кожного запуску r записано історію запусків за допомогою послідовності пар $H_r = \{(f_i, t_i)\}$, де f_i — значення цільової функції, t_i — час знаходження f_i алгоритмом перший раз. Усього в обчислювальному експерименті зроблено 2560 незалежних історій запусків. Для оцінки середнього часу знаходження f_t за допомогою портфеля з N ($N = 1, 8, 16, 32$) алгоритмів вибрано N історій запусків $H_1, \ldots, H_N$ і обчислено час роботи портфеля як

$$t_N = \min_{1, \ldots, N} [\min \{t_i | (f_i, t_i) \in H_r, f_i \geq f_t\}].$$

Для врахування випадковості оцінено середній час знаходження заданого значення цільової функції $E[t_N(f_t)]$ на основі пробних вибірок з 2560 незалежних історій запуску. Наприклад, для оцінки середнього часу знаходження цільової функції для portfolio-8 неодноразово відбирали вісім історій і обчислювали $t_N(f_t)$ для всіх значень f_t. Використано 10000 пробних вибірок для всіх оцінок.

Для команд алгоритмів зібрано результати з 320 запусків team-8, 160 запусків team-16 і 80 запусків team-32. Аналогічно експериментам з портфелями алгоритмів записано історії запусків $H = \{(f_i, t_i)\}$ та обчислено середній час знаходження заданих значень цільової функції. Зазначимо, що показники ефективності команд не можна оцінювати за даними окремих запусків, тому що зв'язок руйнує припущення про незалежність, і немає можливості використати дані експериментів з портфелями алгоритмів.

Ефективність команд алгоритмів team-8x16, team-16x8 і team-32x4 оцінено за результатами історії запусків team-8, team-16 і team-32 відповідно. Розрахунки виконано аналогічно наведеним вище розрахункам для портфелів, оскільки кожну команду можна розглядати як окремий алгоритм.

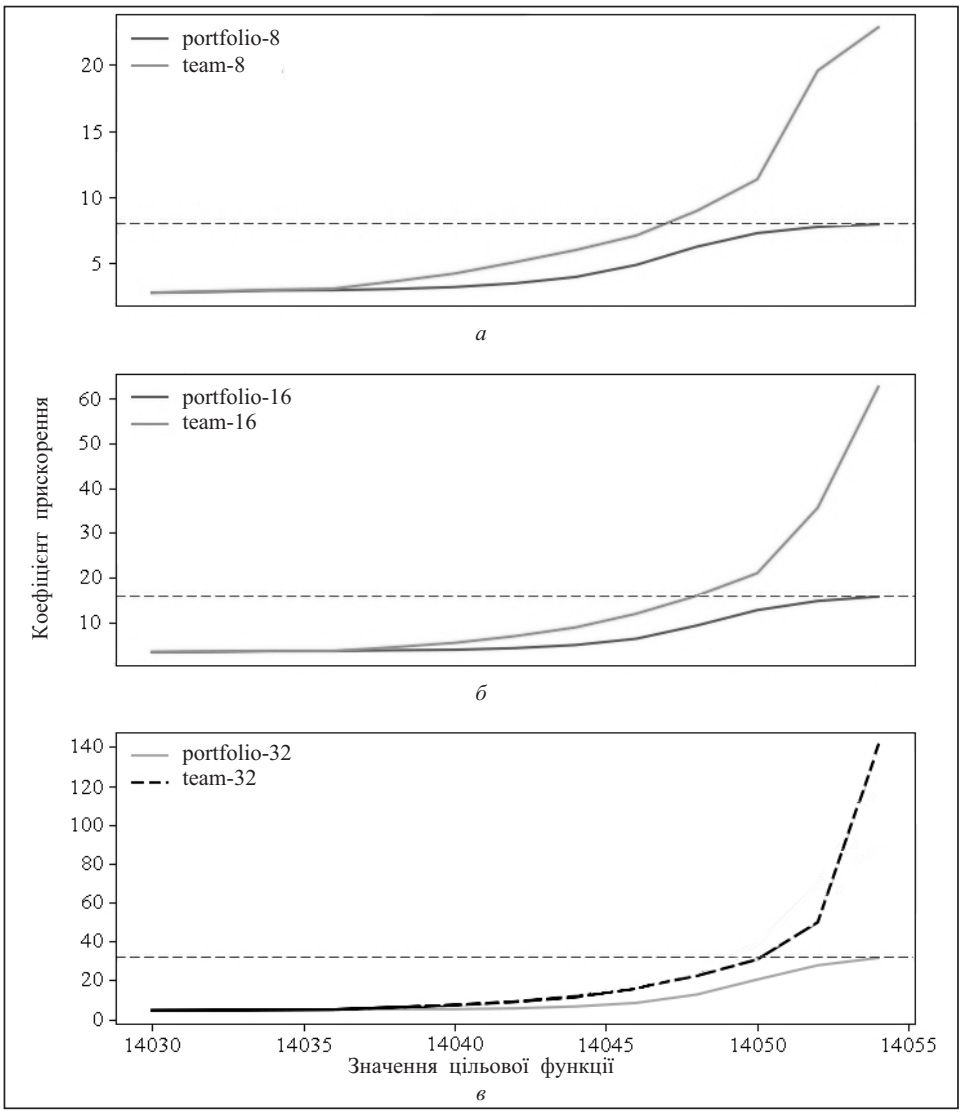

Рис. 2. Графіки прискорення середнього часу знаходження значення цільової функції за допомогою портфелів і команд алгоритмів порівняно з одним алгоритмом

Деякі результати обчислювального експерименту представлено на рис. 2. На рис. 2, *а* для портфеля з восьми алгоритмів та команди з восьми алгоритмів наведено коефіцієнт паралельного прискорення, розрахований відносно портфеля з одного алгоритму. Аналогічну картину можна спостерігати для портфелів і команд з кількістю алгоритмів 16 і 32 (рис. 2, *б, в*). Пунктирні лінії відповідають лінійному прискоренню (8, 16 і 32). Зазначимо, що конфігурація команди team-32 досягає 140-кратного коефіцієнта прискорення, який значно перевищує лінійний коефіцієнт. Отже, командний підхід забезпечує надлінійне прискорення пошуку високоякісних розв'язків, а портфель алгоритмів забезпечує коефіцієнт прискорення, який наближається до лінійного коефіцієнта прискорення.

ВИСНОВКИ

Проведені дослідження свідчать про те, що використання об'єднань алгоритмів дає змогу суттєво прискорити оптимізаційний процес. З аналізу їхніх результатів видно, що особливо відчутним є ефект прискорення у разі впровадження обміну інформацією між алгоритмами команд порівняно з відомими результатами для портфелів алгоритмів. Командний підхід забезпечує надлінійне прискорення пошуку високоякісних розв'язків, а коефіцієнт прискорення портфеля алгоритмів наближається до лінійного коефіцієнта прискорення. Отримані результати свідчать про перспективність досліджень щодо розвитку і вдосконалення команд алгоритмів дискретної оптимізації. Це стосується, зокрема, розроблення та модифікації алгоритмів для створення команд, побудови нових схем обміну інформацією, керувальних програм для команд, складу даних, якими будуть обмінюватись алгоритми. До того ж, звісно, виникає потреба у збільшенні кількості алгоритмів у командах, проведенні експериментальних розрахунків на багатопроцесорних обчислювальних комплексах.

СПИСОК ЛІТЕРАТУРИ

1. Sergienko I.V. Topical directions of informatics. In memory of V. M. Glushkov. New York; Heidelberg; Dordrecht; London: Springer, 2014. 286 p. https://doi.org/10.1007/978-1-4939-0476-1.
2. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. Москва: Мир, 1985. 512 с.
3. Михалевич В.С., Трубин В.А., Шор Н.З. Оптимизационные задачи производственно-транспортного планирования. Москва: Наука, 1986. 264 с.
4. Glover F., Laguna M. Tabu search. Boston: Kluwer Academic Publishers, 1997. 382 p.
5. Сергиенко И.В., Шило В.П. Задачи дискретной оптимизации: проблемы, методы решения, исследования. Киев: Наук. думка, 2003. 264 с.
6. Сергієнко І.В., Шило В.П., Рощин В.О. Дискретна оптимізація. Алгоритми та їхнє ефективне використання. Київ: Наук. думка, 2020. 144 с.
7. Стоян Ю.Г., Яковлев С.В., Пичугина О.С. Евклидовы комбинаторные конфигурации. Харьков: Константа, 2017. 404 с.
8. Семенова Н.В., Колєчкіна Л.М. Векторні задачі дискретної оптимізації на комбінаторних множинах: Методи дослідження та розв'язання. Київ: Наук. думка, 2009. 266 с.
9. Souravlias D., Parsopoulos K., Kotsireas I., Pardalos P. Algorithm portfolios: Advances, applications, and challenges. New York: Springer, 2021. 108 p. https://doi.org/10.1007/978-3-030-68514-0.
10. Максишко Н.К., Заховалко Т.В. Моделі та методи розв'язання прикладних задач покриття на графах та гіперграфах. Запоріжжя: Поліграф, 2009. 244 с.
11. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації. Київ: ВПЦ «Київський університет», 2016. 144 с.
12. Козин И.В. Принципы симметрии в теории принятия решений. Запорожье: Полиграф, 2007. 164 с.

13. Гупал А.М., Сергиенко И.В. Симметрия в ДНК. Методы распознавания дискретных последовательностей. Киев: Наук. думка, 2016. 228 с.
14. Mostovyi O., Prokopyev O.A., Shylo O.V. On maximum speedup ratio of restart algorithm portfolios. *INFORMS J. on Computing.* 2013. Vol. 25, N 2. P. 222–229.
15. Luby M., Ertel W. Optimal parallelization of Las Vegas algorithms. *Lecture Notes in Computer Science*. 1994. Vol. 775. P. 461–474.
16. Shylo O.V., Middelkoop T., Pardalos P.M. Restart strategies in optimization: Parallel and serial cases. *Parallel Computing*. 2011. Vol. 37, N 1. P. 60–68.
17. Gomes C.P., Selman B. Algorithm portfolios. *Artificial Intelligence.* 2001. Vol. 126, N 1–2. P. 43–62.
18. Huberman B.A. An economics approach to hard computational problems. *Science*. 1997. Vol. 275, N 5296. P. 51–54.
19. Chabrier A., Danna E., Pape C.L., Perron L. Solving a network design problem. *Annals of Oper. Res.* 2004. Vol. 130, N 1–4. P. 217–239.
20. Шило В.П., Рощин В.А., Шило П.В. Построение портфеля алгоритмов для распараллеливания процесса решения задачи о максимальном взвешенном разрезе графа. *Компьютерная математика*. 2014. № 2. С. 163–170.
21. Shylo O.V., Prokopyev O.A., Rajgopal J. On algorithm portfolios and restart strategies. *Oper. Res. Lett*. 2011. Vol. 39, N 1. P. 49–52.
22. Shylo V.P., Shylo O.V. Path relinking scheme for the max-cut problem within global equilibrium search. *International J. of Swarm Intelligence Res*. 2011. Vol. 2, N 2. P. 42–51.
23. Shylo V.P., Glover F., Sergienko I.V. Teams of global equilibrium search algorithms for solving the weighted maximum cut problem in parallel. *Cybernetics and Systems Analysis*. 2015. Vol. 51, N 1. P. 16–24. https://doi.org/10.1007/s10559-015-9692-2.
24. Barahona F., Grotschel M., Junger M., Reinelt G. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*. 1988. Vol. 36, N 3. P. 493–513.
25. Chang K.C., Du D.H.-C. Efficient algorithms for layer assignment problem. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*. 1987. Vol. 6, Iss. 1. P. 67–78.
26. Poljak S., Tuza Z. Maximum cuts and large bipartite subgraphs. In: Combinatorial Optimization. AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Cook W., Lovasz L., Seymour P. (Eds.). 1995. Vol. 20. P. 181–244.
27. Palubeckis G. Iterated tabu search for the unconstrained binary quadratic optimization problem. *Informatica*. 2006. Vol. 17, N 2. P. 279–296.
28. Helmberg C., Rendl F. A spectral bundle method for semidefinite programming. *SIAM J. on Optimizatio*n. 2000. Vol. 10, Iss. 3. P. 673–696.
29. Glover F., Laguna M., Marti R. Fundamentals of scatter search and path relinking. *Control and Cybernetics*. 2000. Vol. 29, N 3. P. 653–684.
30. Benlic U., Hao J.K. Breakout local search for the max-cut problem. *Engineering Applications of Artificial Intelligence*. 2013. Vol. 26, Iss. 3. P. 1162–1173.

I.V. Sergienko, V.P. Shylo, V.O. Roshchyn

ALGORITHM UNIONS FOR SOLVING DISCRETE OPTIMIZATION PROBLEMS

Abstract. The algorithm unions (portfolios and teams) of optimization algorithms, their properties, and their impact on the acceleration of the optimization process are considered. The global equilibrium search applications in algorithm unions to individual discrete optimization problems and various information exchange schemes in algorithm teams are researched. Significant emphasis is placed on the experimental research and analysis of the developed algorithm portfolios and teams, conducted sequentially and using the multiprocessor computing complex SCIT-4 of the V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine. The resulting efficiency of algorithm unions shows that the team approach has significant advantages over algorithm portfolios.

Keywords: algorithm unions (portfolios and teams), discrete optimization, information exchange, experimental research, algorithm unions efficiency.

Надійшла до редакції 13.03.2023